Visual Grid Programming For Application Scientists

Keith R. Jackson
Lawrence Berkeley National Laboratory

Problem

- Current interfaces to Grid and Collaborative technologies are too complex for most application scientists.
  – Very steep learning curve.
    - Often times perceived as easier to develop a one-off solution then to adopt standard middleware tools.
  – Provides a major barrier to wide-spread adoption.
- Poor integration between collaborative tools and Grid middleware.

High-Level Goals

- Provide tools that the scientific community is already familiar with.
- Provide high-level interfaces that capture common scientific tasks.
- Integrate collaborative tools directly.
- Recognize that application scientists have a wide variety of CS knowledge.
  - Expert users who can program their own tools.
  - Sophisticated users that can extend existing tools.
  - Normal users of tools.
- No single solution will be appropriate for all situations!

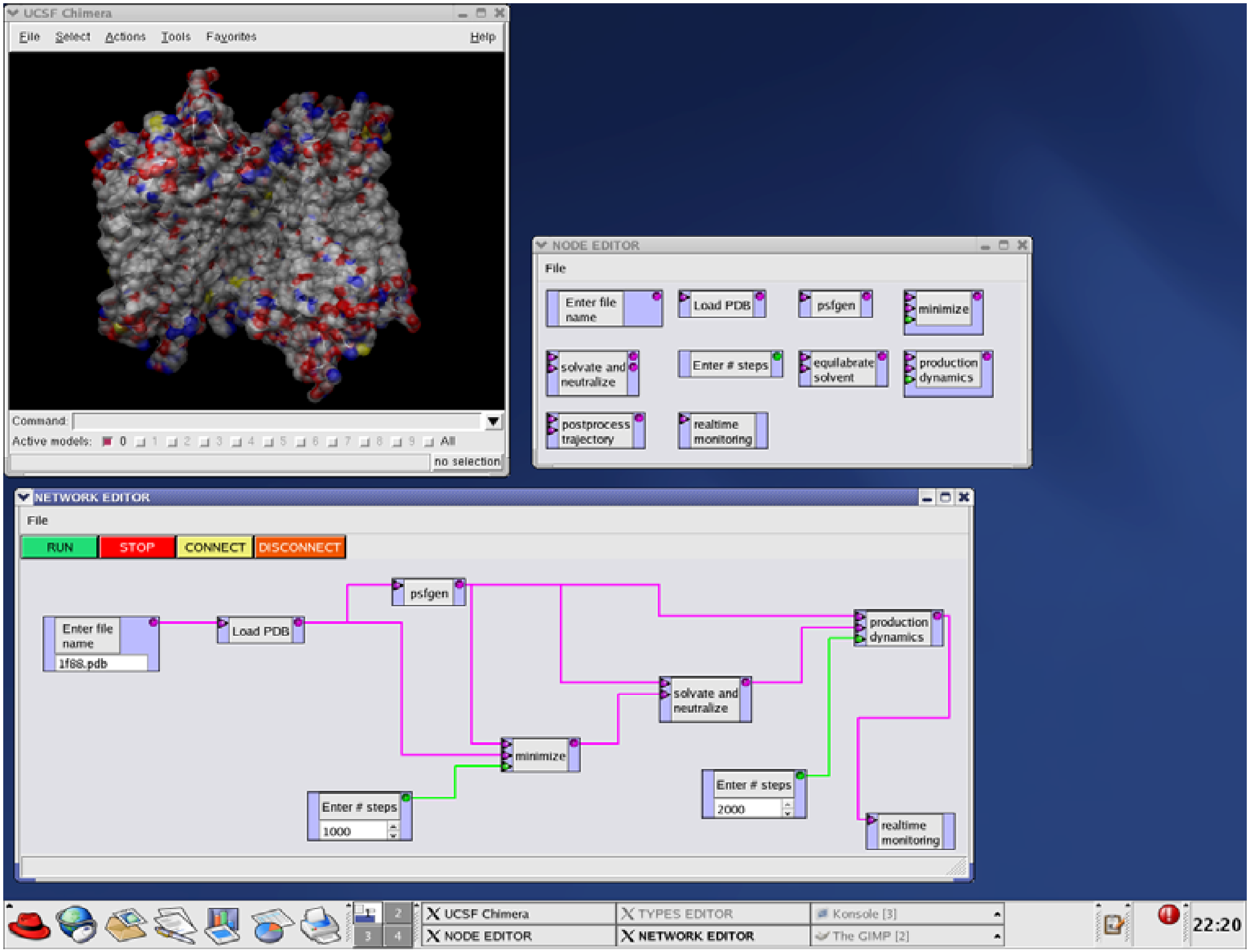
High-Level Abstractions

- What have we tried?
  - High-Level programming languages and abstractions
    - Useful for expert users, but too complex for many.
    - Domain specific abstractions increase the usability for moderately sophisticated users.
    - An important component for developing Visual Programming environments.
  - Portals
    - Very useful for providing easy to use interfaces for many users.
    - Excellent for collaboration.
    - Can be very difficult to customize for the sophisticated users.
    - It is very challenging to do highly interactive interfaces.

Visual Programming (VP)

- For many scientific applications it is natural to model their process visually.
  - Can easily drag and drop components to construct a variety of different experiments.
    - Networks can be shared amongst collaborators.
  - Need to support both modeling data and control flows.
- Many scientists are already familiar with this model from tools like AVS®.

Example VP Environment



VP Requirements

- Implement in a high-level, interpreted, simple to use language that has support in the scientific community.
- Must integrate with the next generation of Grid Services, e.g., WS-RF.
  - WS-RF makes introspection for service composition significantly easier.
    - This allows for late binding which is vitally important in a dynamic Grid environment.

VP Requirements

- Support the integration of collaborative technologies to allow for shared visual development.
  - Could be integrated in a portal, or used with the Access Grid, etc.
- Use high-level abstractions to allow submission to multiple execution engines.
  - Direct Globus submissions.
  - DAGMan from the Condor Project.
  - BPEL4WS, or other emerging standards.

Why Interpreted Languages

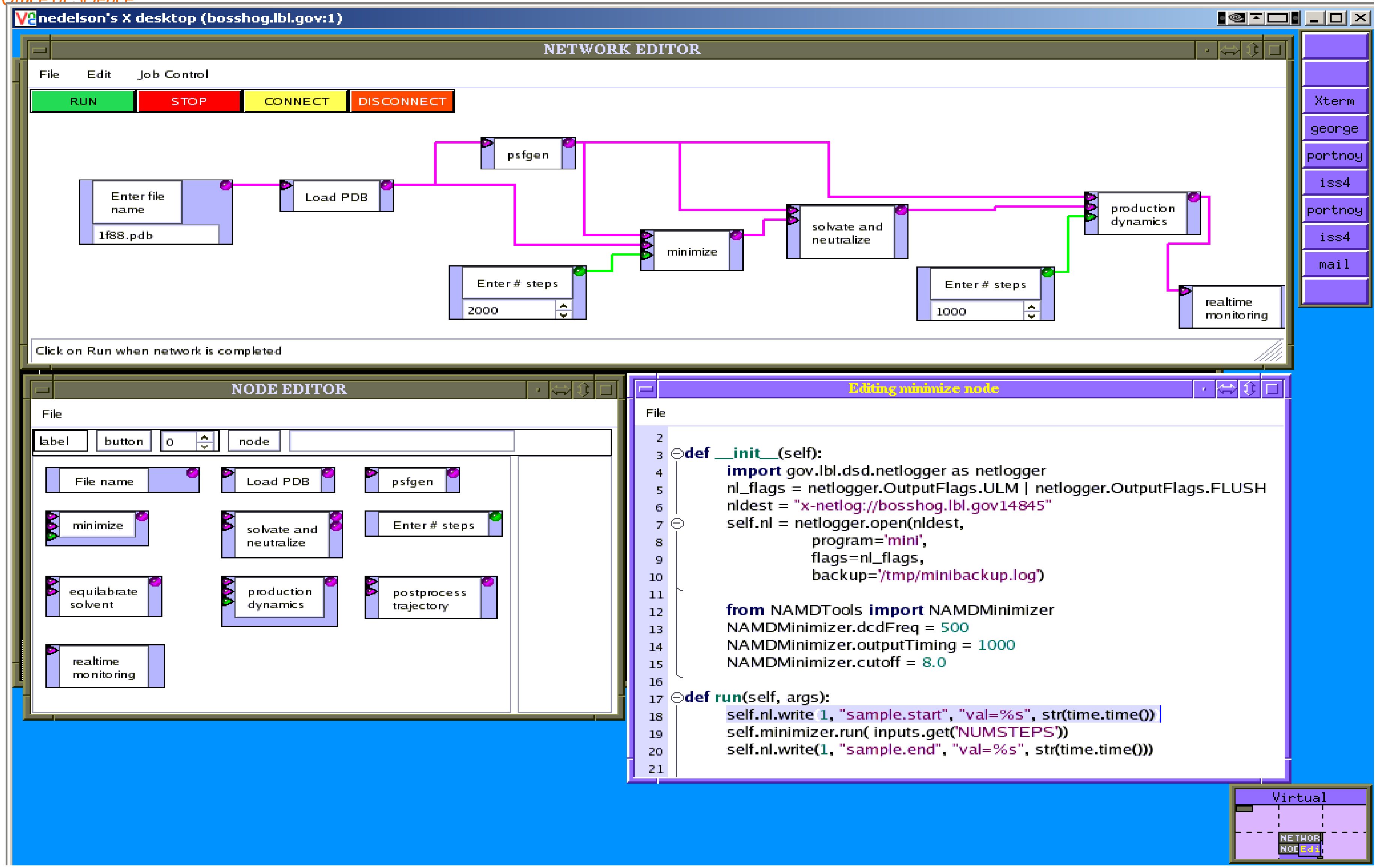
- Easier to support the three classes of users mentioned earlier.
  - Expert application scientists will customize the general framework for their domain.
    - Interpreted languages significantly lower the barrier to doing this by shortening development and debug time.
  - Sophisticated users will be able to customize the domain specific tool.
    - Interpreted languages support the ability to dynamically edit visual nodes on the fly using simplified interfaces.
- Well suited to integrating existing components.
  - Faster and easier to develop with.

U.S. Department of Energy

Office of Science

Dynamic Customization

BERKELEY LAB



Conclusions

- Research is necessary to develop better ways to allow scientists to interact with distributed Grid Services.
- No one solution will fit all users.
- Grid programming environments must support the collaborative nature of science today.

Conclusions

- A properly designed Visual Programming environment can meet the needs of many application scientists.
  - Requires the CS community to develop generic frameworks that can easily be customized by expert application scientists for their specific domain.
  - Must also support the ability of non-expert programmers to modify the domain specific environment to fit their specific needs.